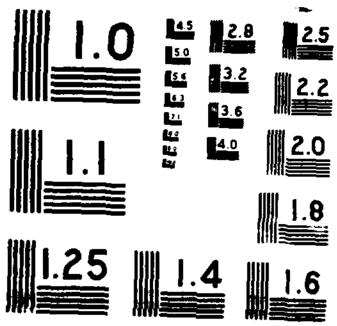
AD-A206 866 PROGRAMS TO SWAP DIAGONAL BLOCKS(U) CALIFORNIA UNIV BERKELEY CENTER FOR PURE AND APPLIED MATHEMATICS K C NG ET AL. JUN 87 PAM-381 N00014-85-K-0180 1/1

UNCLASSIFIED F/G 12/2 NL

END
FILMED
5-89
DTIC


1.0
4.5
5.0
2.8
2.5
3.2
2.2
3.6
4.0
1.1
2.0
1.8
1.25
1.4
1.6

2

PROGRAMS TO SWAP DIAGONAL BLOCKS

BY

K.C. Ng[1] and B.N. Parlett[2]

May 21, 1987

ABSTRACT

The real Schur form of a real square matrix is block upper triangular. We study techniques for performing orthogonal similarity transformations that preserve block triangular form but alter the order of the eigenvalues along the (block) diagonal. Keywords: Software economizer, implementation, Cholesky factor, numerical tests.

[1] Sun Microsystems, Inc.

[2] Center for Pure and Applied Mathematices, Univ. of California, Berkeley, CA 94720.

This work was partially supported by Office of Naval Research Contract No. N00014-85-K-0180

89 4 18 055

1. Introduction

A triangular matrix reveals its eigenvalues along the main diagonal. By Schur's lemma any square complex matrix is unitarily similar to an upper triangular matrix with the eigenvalues arranged in any desired order along the main diagonal. It follows that any real square matrix is othogonally similar to a real block upper triangular matrix in which each 2×2 block on the diagonal corresponds to a pair of complex conjugate eigenvalues. The Householder-QR algorithm is a stable, efficient algorithm that produces a Schur form. However the ordering of the eigenvalues that the QR algorithm produces may not be suitable for certain purposes, such as computing the exponential of the original matrix. There are programs in the library EISPACK that compute this real Schur form. See section 2.3.6 of [EIS,1976].

This investigation presents and compares all the attractive methods we can think of for performing orthogonal similarity transformations that preserve block triangular form but rearrange the eigenvalues. This is a fairly straightforward task but it is always a challenge to try and keep down three conflicting costs: round off error, execution time, and program length.

We give some attention to the task of swapping adjacent diagonal blocks of orders p and q but our main concern is with the case p=q=2. We use capital letters to denote matrices. Fortran programs are given at the end.

Before plunging into details we describe the methods in brief general terms. **Algorithm 0** (G.W. Stewart): Swap adjacent blocks using one or two QR steps with a pre-determined shift to force the ordering of the eigenvalues of the new blocks. **Algorithm 1**: Swap adjacent blocks as needed using an explicit orthogonal similarity transformation. At most 4 rows and columns will be modified at each swap. **Algorithm 2**: Swap adjacent blocks using Householder transformations. For swapping a pair of 2×2 blocks two Householder transformations are needed.

The table below compares the algorithms for code length and running time. THe remainder of that paper is concerned with accuracy.

Algorithm number	Fortran line count	Speed ratio (The ratio was determined by runs on 9×9 matrices)
0	165	1.27
1	386	1.00
2	301	1.15

Table 1

We were encouraged to present our programs and results by Dr. Sven Hammarling (of NAG, Inc., Oxford) who showed us work on swapping that he had begun in cooperation with Dr. J. Dongarra (Argonne National Laboratory) and the late Prof. J. H. Wilkinson.

2. The Software Economizer (EXCHNG)

Consider a submatrix of the form

$$\begin{bmatrix} A_1 & B \\ 0 & A_2 \end{bmatrix}$$

where A_1 and A_2 are 2×2 diagonal blocks.

Algorithm 0 (called EXCHNG in [Ste.,1976]).

1. An implicit double shift is determined from A_1.
2. An arbitrary QR step is performed to destroy the triangular form and put the matrix into Hessenberg form.
3. A sequence of double QR steps using the shift from step 1. The The eigenvalues of the first block will emerge in the lower part of the array occupied by both blocks, usually in one or two steps.

Remark 1. The alogrithm discards the information that there are two pairs of conjugate complex eigenvalues. Stewart modifies the standard QR program so that a supplied initial shift may overwrite the usual Francis shift at the first step. Such an algorithm would converge in one or two steps.

3. General Theory

Consider the block upper triangular matrix

$$\begin{bmatrix} A_1 & B \\ 0 & A_2 \end{bmatrix}, \quad \begin{array}{l} A_1 \text{ is } p\times p, \\ A_2 \text{ is } q\times q. \end{array} \tag{1}$$

Throughout this paper we assume that A_1 and A_2 have no eigenvalue in common. It follows that there exists a unique $p\times q$ matrix X such that

$$A_1X - XA_2 = B. \tag{2}$$

This is called Sylvester's equation. It follows that

$$\begin{bmatrix} A_1 & B \\ 0 & A_2 \end{bmatrix} = \begin{bmatrix} I_p & -X \\ 0 & I_q \end{bmatrix} \cdot \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} \cdot \begin{bmatrix} I_p & X \\ 0 & I_q \end{bmatrix},$$

$$= \begin{bmatrix} -X & I_p \\ I_q & 0 \end{bmatrix} \cdot \begin{bmatrix} A_2 & 0 \\ 0 & A_1 \end{bmatrix} \cdot \begin{bmatrix} 0 & I_q \\ I_p & X \end{bmatrix}. \tag{3}$$

DEFINITION. An orthogonal $(p+q)\times(p+q)$ matrix H is said to **swap** A_1 and A_2 if

$$H \cdot \begin{bmatrix} A_1 & B \\ 0 & A_2 \end{bmatrix} \cdot H^T = \begin{bmatrix} \tilde{A}_2 & \tilde{B} \\ 0 & \tilde{A}_1 \end{bmatrix} \tag{4}$$

where $\tilde{A}_i$ is similar to A_i, $i=1,2$.

Lemma 1. An orthogonal $(p+q)\times(p+q)$ matrix H swaps A_1 and A_2 if, and only if,

$$H \cdot \begin{bmatrix} -X \\ I_q \end{bmatrix} = \begin{bmatrix} M_2 \\ 0 \end{bmatrix}, \tag{5}$$

for some invertible $q\times q$ M_2 where X is defined in (2).

Note that, since H is invertible,

$$\operatorname{rank}\begin{bmatrix} M_2 \\ 0 \end{bmatrix} = \operatorname{rank}\begin{bmatrix} -X \\ I_q \end{bmatrix} = q.$$

Consequently M_2 is $q\times q$ and must be invertible.

Proof. If H satisfies (5) then for some $q \times p$ W and $p \times p$ M_1,

$$H \cdot \begin{bmatrix} -X & I_p \\ I_q & 0 \end{bmatrix} = \begin{bmatrix} M_2 & W \\ 0 & M_1 \end{bmatrix}$$

and, since both matrices on the left are invertible so are M_1 and M_2. Thus

$$H \cdot \begin{bmatrix} A_1 & B \\ 0 & A_2 \end{bmatrix} \cdot H^T = H \cdot \begin{bmatrix} -X & I_p \\ I_q & 0 \end{bmatrix} \cdot \begin{bmatrix} A_2 & 0 \\ 0 & A_1 \end{bmatrix} \cdot \begin{bmatrix} 0 & I_q \\ I_p & X \end{bmatrix} \cdot H^T,$$

$$= \begin{bmatrix} M_2 & W \\ 0 & M_1 \end{bmatrix} \cdot \begin{bmatrix} A_2 & 0 \\ 0 & A_1 \end{bmatrix} \cdot \begin{bmatrix} M_2^{-1} & -M_2^{-1} \cdot W \cdot M_1^{-1} \\ 0 & M_1^{-1} \end{bmatrix},$$

$$= \begin{bmatrix} \tilde{A}_2 & \tilde{B} \\ 0 & \tilde{A}_1 \end{bmatrix},$$

where

$$\tilde{A}_i = M_i \cdot A_i \cdot M_i^{-1}, \ i=1,2; \quad \tilde{B} = (WA_1 - \tilde{A}_2 W) \cdot M_1^{-1}.$$

Conversely, if H swaps A_1 and A_2 then there exist M_1, M_2, and W such that

$$\begin{bmatrix} \tilde{A}_2 & \tilde{B} \\ 0 & \tilde{A}_1 \end{bmatrix} = \begin{bmatrix} M_2 & W \\ 0 & M_1 \end{bmatrix} \cdot \begin{bmatrix} A_2 & 0 \\ 0 & A_1 \end{bmatrix} \cdot \begin{bmatrix} M_2^{-1} & -M_2^{-1} \cdot W \cdot M_1^{-1} \\ 0 & M_1^{-1} \end{bmatrix},$$

$$= H \cdot \begin{bmatrix} -X & I_p \\ I_q & 0 \end{bmatrix} \cdot \begin{bmatrix} A_2 & 0 \\ 0 & A_1 \end{bmatrix} \cdot \begin{bmatrix} 0 & I_q \\ I_p & X \end{bmatrix} \cdot H^T.$$

It follows that

$$D \equiv \begin{bmatrix} M_2 & W \\ 0 & M_1 \end{bmatrix}^{-1} \cdot H \cdot \begin{bmatrix} -X & I_p \\ I_q & 0 \end{bmatrix} \quad \textbf{commutes with} \quad \begin{bmatrix} A_2 & 0 \\ 0 & A_1 \end{bmatrix}.$$

Since A_1 and A_2 have no eigenvalues in common D must be a polynomial in diag(A_2, A_1). See [Gant. vol. 1, p.222].

$$H \cdot \begin{bmatrix} -X & I_p \\ I_q & 0 \end{bmatrix} = \begin{bmatrix} M_2 & W \\ 0 & M_1 \end{bmatrix} \cdot D$$

must be block upper triangular. This establishes the converse.

QED

Lemma 2. An othogonal H that swaps A_1 and A_2 must have the form

$$H = \begin{bmatrix} C_2^{-1} & 0 \\ 0 & C_1^{-1} \end{bmatrix} \cdot \begin{bmatrix} -X^T & I_q \\ I_p & X \end{bmatrix} \tag{6}$$

where

$$\begin{aligned} C_2 \cdot C_2^T &= I_q + X^T \cdot X \ , \\ C_1 \cdot C_1^T &= I_p + X \cdot X^T \ . \end{aligned} \tag{7}$$

Proof. Write C_2^T for M_2, multiply (5) by H^T and use the orthogonality of H to find

$$\begin{bmatrix} -X \\ I_q \end{bmatrix} = H^T \cdot H \cdot \begin{bmatrix} -X \\ I_q \end{bmatrix} = H^T \cdot \begin{bmatrix} C_2^T \\ 0 \end{bmatrix} = H^T \cdot \begin{bmatrix} I_q \\ 0 \end{bmatrix} \cdot C_2^T \ .$$

Transposing reveals the first row of H. The second row follows by orthogonality.

QED

Remark 1. There are infinitely many choices for C_1 and C_2 that satisfy (7). See Section 4.3 for more details.

Remark 2. One of our implementations uses the form in (6) explicitly. The block rows are orthogonal by their form, so it is the accuracy with which (7) is fulfilled that determines the orthogonality of the computed H. An alternative implementation starts from (5) and seeks H as a product of elementary reflectors (also known as Householder matrices).

The key blocks of the transformed matrix can be found explicitly. Using (3), (6), and (7) it is not difficult to see that

$$\begin{aligned} \tilde{A}_2 &= C_2^T \cdot A_2 \cdot C_2^{-T} , \quad \tilde{A}_1 = C_1^{-1} \cdot A_1 \cdot C_1 \ , \\ \tilde{B} &= C_2^T \cdot A_2 \cdot X^T \cdot C_1^{\ T} - C_2^{-1} \cdot X^T \cdot A_1 \cdot C_1 \ . \end{aligned} \tag{8}$$

4. Implementation details

4.1. Standardized Real Schur Form

The Schur form of a matrix is not unique and the real Schur form of a real matrix offers even more freedom. We urge the adoption of the following conventions.

i) 2×2 diagonal blocks are used exclusively for complex conjugate pairs of eigenvalues, not for distinct real eigenvalues.

ii) The diagonal elements of 2×2 diagonal blocks are made equal. This value is the real part of each eigenvalue.

Consequently we advocate the form

$$\begin{bmatrix} \alpha & \beta \\ \gamma & \alpha \end{bmatrix}, \quad \beta\cdot\gamma < 0 .$$

The off diagonal elements of the 2×2 diagonal blocks cannot always be made equal in absolute value but they must be opposite in sign. To guarant uniqueness one may require β and γ to satisfy $0 < \gamma \le -\beta$, but that is not essential. Note that the eigenvalues are $\alpha \pm \sqrt{\beta\cdot\gamma}$.

The use of a *standard* real Schur form facilitates the swapping of diagonal blocks as well as ensuring that the real parts of all eigenvalues are held on the diagonal of the real Schur form.

If a given real Schur form does not have its eigenvalues ordered appropriately down the diagonal then some swapping of diagonal blocks will be needed. However the task is considerably simplified by the fact that no block has order exceeding 2. Any configuration of eigenvalues can be reached by swapping adjacent diagonal blocks and this is the task we consider below.

Here is a method (cf section 4.5) to put 2×2 diagonal blocks into standard form. Let

$$A = \begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix}.$$

Define a reflection P by

$$P^T = P = \begin{bmatrix} -\cos(\vartheta) & \sin(\vartheta) \\ \sin(\vartheta) & \cos(\vartheta) \end{bmatrix}, \quad \vartheta = \frac{1}{2}\tan^{-1}\left(\frac{a11 - a22}{a12 + a21}\right).$$

Write $c=\cos(\theta)$ and $s=\sin(\theta)$. It is not difficult to see that PAP transforms A to a standard form:

$$PAP = \begin{bmatrix} \frac{a11+a22}{2} & a21 - \frac{c}{s}\cdot\frac{a11-a22}{2} \\ a12 - \frac{c}{s}\cdot\frac{a11-a22}{2} & \frac{a11+a22}{2} \end{bmatrix}. \qquad (9)$$

4.2. Solving $A_1X - XA_2 = B$

Considerable attention has been paid to the general case of this equation, now known as Sylvester's equation. See [B&S,1972] and [G,N,& vL,1979]. When A_1 and A_2 are either 1×1 or standardized 2×2 matrices the solution can be given explicitly using stable formulae.

In an earlier unpublished report [Pa,1977] we advocated scaling X, i.e., we solved

$$A_1X - XA_2 = \xi\cdot B$$

with ξ chosen so that $\|X\|\cong 1$. Further analysis shows that this caution is unnecessary. There is no danger in working with X of large norm **provided** that $\|X\|^2$ does not overflow. Moreover if $\|X\|^2$ does overflow then the blocks should not be swapped because a tiny pertubation will give the new A_1 and A_2 at least one common eigenvalue.

Our algorithm for solving the Sylvester equation is called TXMXT (for TX-XT) and is described in Appendix A, see also the program Appendix C.

4.3. The Choleski Factor

If the explicit orthogonal H described in Section 3 is to be used then it is necessary to solve the equations (7) for C_1 and C_2. We can see no reason to avoid the Choleski factorization. The formulae are given below. When presenting the algorithm in detail we write xij for X(i,j). Recall equation (7):

$$C_2\cdot C_2^T = I_q + X^T\cdot X\ ,$$

$$C_1\cdot C_1^T = I_p + X\cdot X^T\ .$$

Algorithm 1.

An H is found explicitly in the form of (6) to swap A_1 and A_2. The choice for C_1 and C_2 in (7) are the Choleski factors.

Case 1. X is 1×1, then $C_2(1,1) = C_1(1,1) = \sqrt{1+x11^2}$.

Case 2. X is 1×2, then $C_1(1,1) = \sqrt{1+x11^2+x12^2}$, and

$$C_2 = \begin{bmatrix} \gamma & 0 \\ \frac{x11 \cdot x12}{\gamma} & \sqrt{1+(\frac{x12}{\gamma})^2} \end{bmatrix}, \quad \gamma = \sqrt{1+x11^2}.$$

Case 3. X is 2×1, then $C_2(1,1) = \sqrt{1+x11^2+x21^2}$, and

$$C_1 = \begin{bmatrix} \gamma & 0 \\ \frac{x11 \cdot x21}{\gamma} & \sqrt{1+(\frac{x21}{\gamma})^2} \end{bmatrix}, \quad \gamma = \sqrt{1+x11^2}.$$

Case 4. X is 2×2, let $\delta = x11 \cdot x22 - x12 \cdot x21$, $\gamma = \sqrt{1+x11^2+x12^2}$, and $\kappa = \sqrt{1+x11^2+x21^2}$; we have

$$C_1 = \begin{bmatrix} \gamma & 0 \\ \frac{x11 \cdot x21 + x12 \cdot x22}{\gamma} & \sqrt{1 + \frac{x21^2+x22^2+\delta^2}{\gamma^2}} \end{bmatrix},$$

and

$$C_2 = \begin{bmatrix} \kappa & 0 \\ \frac{x11 \cdot x12 + x21 \cdot x22}{\kappa} & \sqrt{1 + \frac{x12^2+x22^2+\delta^2}{\kappa^2}} \end{bmatrix}.$$

4.4. Representing H as a product of two reflectors

The explicit form of H in Section 3 is not mandatory. W. Kahan suggested using two reflections instead. Here are the details. First some notation: A n×n reflection (or Householder matrix) can be represented as $I - uu^T/d$, where I is the n×n identity matrix, u is a

n-vector, and $d = \frac{1}{2}\|\mathbf{u}\|^2$. We use the fact that if $\mathbf{u}=\mathbf{x}+\mathbf{y}$ and $\|\mathbf{x}\|=\|\mathbf{y}\|$, then $(\mathbf{I}-\mathbf{u}\mathbf{u}^T/(\frac{1}{2}\|\mathbf{u}\|^2))\cdot\mathbf{x}=-\mathbf{y}$.

Algorithm 2.

An **H** is found implicitly in the form of either a reflector or a product of two reflectors to swap A_1 and A_2. The reflector(s) are determined as follows:

Case 1. X is 1×1. Let $sx = \text{sign}(x11)\cdot\sqrt{1+x11^2}$. We seek a reflection H so that

$$H\cdot\begin{bmatrix}-x11\\ 1\end{bmatrix} = \begin{bmatrix}-sx\\ 0\end{bmatrix}.$$

The special form of H leads to

If $x11/sx \le 0.5$, then $u1 = sx - x11$; else $u1 = 1/(sx+x11)$
$u2 = 1$
$d = u1\cdot sx$

Case 2. X is 1×2. Let $sx = -\text{sign}(x12)\cdot\sqrt{1+x11^2+x12^2}$. We observe that if a reflector H satisfies

$$H\cdot\begin{bmatrix}1\\ x11\\ x12\end{bmatrix} = \begin{bmatrix}0\\ 0\\ -sx\end{bmatrix},$$

then it satisfies (5). The proof is left to the reader. The special form of H leads to

$u1 = 1$
$u2 = x11$
if $-x12/sx \le 0.5$, then $u3 = x12+sx$; else $u3 = (1+x11^2)/(sx-x12)$
$d = u3\cdot sx$

Case 3. X is 2×1. Let $sx = \text{sign}(x11)\cdot\sqrt{1+x11^2+x21^2}$. From (5) we seek a reflection H so that

$$H\cdot\begin{bmatrix}-x11\\ -x21\\ 1\end{bmatrix} = \begin{bmatrix}-sx\\ 0\\ 0\end{bmatrix}.$$

The special form of H leads to

If $x11/sx \le 0.5$, then $u1 = sx - x11$; else $u1 = (1+x21^2)/(sx+x11)$
$u2 = -x21$
$u3 = 1$
$d = u1 \cdot sx$

Case 4. X is 2×2. Two reflections H_1 and H_2 are required. In (5) let M_2 be upper triangular and $H = H_2 \cdot H_1$. First define

$$sx = \text{sign}(x) \cdot \sqrt{1+x11^2+x21^2}.$$

We seek a reflection $H_1 = I - \mathbf{u}\mathbf{u}^T/d$ so that

$$H_1 \cdot \begin{bmatrix} -x11 \\ -x21 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -sx \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

From the special form of H we have

If $x11/sx \le 0.5$, then $u1 = sx - x11$; else $u1 = (1+x21^2)/(sx+x11)$
$u2 = -x21$
$u3 = 1$
$u4 = 0$
$d = u1 \cdot sx$.

Next, define an intermediate vector $\mathbf{y}$ by

$$\mathbf{y} = \begin{bmatrix} y1 \\ y2 \\ y3 \\ y4 \end{bmatrix} = H_1 \cdot \begin{bmatrix} -x12 \\ -x22 \\ 0 \\ 1 \end{bmatrix}.$$

One can verify that

$y1 = -(x11 \cdot x12 + x21 \cdot x22)/sx$,
$y2 = -x22 - x21 \cdot (x12 \cdot u1 - x21 \cdot x22)/d$,
$y3 = (x12 \cdot u1 - x21 \cdot x22)/d$,
$y4 = 1$.

Note that $y2 = -x22 - x21 \cdot y3$. Let $sy = -\text{sign}(y2) \cdot \sqrt{1+y2^2+y3^2}$.

We seek the second reflection $H_2 = I - \mathbf{v}\mathbf{v}^T/g$ so that

$$H_2 \cdot \begin{bmatrix} y1 \\ y2 \\ y3 \\ y4 \end{bmatrix} = \begin{bmatrix} y1 \\ -sy \\ 0 \\ 0 \end{bmatrix}.$$

There is no need to change the top row. Proceed as before to obtain

v1 = 0

if -y2/sy ≤ 0.5, then v2 = sy + y2; else v2 = $(1+y3^2)/(sy-y2)$

v3 = y3

v4 = 1

g = v2·sy.

Remark. Since each H that swaps A_1 and A_2 can be represented in the form of (6) (lemma 2), it is worthwhile to see what the reflection, or product of reflections, looks like in this form. In fact we make use of it in Section 4.5. We compute the corresponding C_1 and C_2 in appendix B. They are obtained by noting that

$$H \cdot \begin{bmatrix} -X \\ I_q \end{bmatrix} = \begin{bmatrix} C_2^T \\ 0 \end{bmatrix} \text{ and } \begin{bmatrix} I_p & X \end{bmatrix} \cdot H^T = \begin{bmatrix} 0 & C_1 \end{bmatrix}.$$

4.5. Special treatment for the diagonal blocks.

From (8) the new diagonal block $\tilde{A}$ is equal to $W \cdot A \cdot W^{-1}$ for some W. Given A,W we use a special subroutine (called EQUDI, see Appendix C) to put 2×2 diagonal block $\tilde{A} = W \cdot A \cdot W^{-1}$ into standard form and effect the associated changes in the corresponding rows and columns of the Schur form. For better accuracy we derive here the analytic formulae for the transformation, based on W and A. Recall that a11=a22.

d = det(W),

z = a21·w12·w22 - a12·w11·w21;

$$WAW^{-1} = \begin{bmatrix} a11 + z/d & \dfrac{a12\cdot w11^2 - a21\cdot w12^2}{d} \\ \dfrac{a21\cdot w22^2 - a12\cdot w21^2}{d} & a11 - z/d \end{bmatrix}. \quad (10)$$

Apply (9) in Section 4.1 to $\tilde{A}=WAW^{-1}$ above to find

$$u = a12\cdot(w11-w21)\cdot(w11+w21) + a21\cdot(w22-w12)\cdot(w22+w12)$$

$$\theta = \tfrac{1}{2}\cdot\tan^{-1}(2z/u)$$

$s = \sin(\theta)$, $c = \cos(\theta)$, and

$$P\tilde{A}P = \begin{bmatrix} a11 & v1 \\ v2 & a11 \end{bmatrix}, \quad (11)$$

where

$$P = \begin{bmatrix} -c & s \\ s & c \end{bmatrix},$$

$$v1 = [\ a21\cdot w22\cdot(w22-w12\cdot c/s) - a12\cdot w21\cdot(w21-w11\cdot c/s)]/d,$$

$$v2 = [-a21\cdot w22\cdot(w22+w12\cdot s/c) + a12\cdot w21\cdot(w21+w11\cdot s/c)]/d$$

Since eigenvalues are preserved under similar transformation, we must have $v1\cdot v2=a12\cdot a21$; thus we may recompute v1 from v2 or vice versa, depending on which one is smaller in magnitude.

5. Numerical Tests

We have done extensive testing on matrices with various mixtures of block size. All 3 algorithms perform well in most cases. To investigate more closely the accuracy of Algorithms 0,1, and 2 under extreme conditions, we tested them on three sets of matrices: one with huge B, one with a choice of B so that $|\det(X)| \ll \|X\|^2$, and finally one with fairly close eigenvalues. These tests perform 2×2 block swaps.

How to measure the "correctness" of the computed output is not so easy. Let $\tilde{A}$ be the output matrix P^TAP where P is the orthogonal matrix that accumulates all the transformations that are applied to A. We believe that the only sensible measures for the accuracy of P and $\tilde{A}$ are 1) how close is $P\cdot P^T$ to the identity matrix? and 2) how close is $P\tilde{A}P^T$ to the original matrix A.

Thus our measuring parameters are:

1. $E_P = \| I - PP^T \|$, (12)
2. $E_A = \|A - P\tilde{A}P^T\| / \|A\|$. (13)

E_P is the *orthogonality error* in P; E_A is the norm relative error in $P^T\tilde{A}P$. Out of curiosity we also computed

3. $\varepsilon = \mathrm{Max}\{|\varepsilon_{i,j}|,\ A(i,j) \neq 0\}$ (14)

where $\varepsilon_{i,j} = |(A - P\tilde{A}P^T)(i,j)/A(i,j)|$. This is the worst relative error among the elements of $P\tilde{A}P^T$.

The third parameters make sense only when $A(i,j) \neq 0$, since fill-in (zero elements become non-zero) is unavoidable in recovering A from P and $\tilde{A}$. We should point out that ε is too exigent a measure for the accuracy of P and $\tilde{A}$. It is unreasonable to demand high relative accuracy for tiny elements in $P\tilde{A}P^T$. Nevertheless we found ε helpful in showing subtle differences between good swapping programs. The following results were obtained on a SUN 3/50. P and $\tilde{A}$ are computed solely in single precision arithmetic. However, the error measures are computed in double precision. Only three digits are displayed for the error measures in order to keep the display clean. We have also run our program on a VAX/750 with similar results.

The Fortran program for Algorithm 0 is Stewart's EXCHNG, the Fortran program for Algorithm 1 and 2 are written according to section 4.3 and 4.4 with special formula for the diagonal blocks described in section 4.5. See the listing in Appendix C.

The roundoff unit is $2^{-23} \approx 1.192\mathrm{E}{-7}$ in the following numerical results.

Test matrix I with parameter τ (large B)

$$A(\tau) = \begin{bmatrix} 2 & -87 & -20000\tau & 10000\tau \\ 5 & 2 & -20000\tau & -10000\tau \\ 0 & 0 & 1 & -11 \\ 0 & 0 & 37 & 1 \end{bmatrix}.$$

τ	Algorithm	ϵ see (14)	E_A see (13)	E_P see (12)
1	0	**2.11e-3**	1.45e-6	1.66e-6
	1	2.56e-6	1.62e-7	1.96e-7
	2	2.74e-6	3.57e-7	3.41e-7
2^{-3}	0	**1.25e-4**	4.89e-7	1.05e-6
	1	1.66e-6	1.27e-7	1.62e-7
	2	8.70e-6	4.15e-7	4.79e-7
2^{-6}	0	5.18e-5	4.69e-7	6.97e-7
	1	1.90e-6	2.19e-7	2.32e-7
	2	8.34e-7	1.93e-7	2.14e-7

P and $\tilde{A} = P^T AP$ from algorithm 2 when $A = A(1)$.

$$X = \begin{bmatrix} -57387.61 & 6294.046 \\ -3106.521 & -7298.501 \end{bmatrix},$$

$$P = \begin{bmatrix} -9.999731E-1 & 7.337808E-3 & -1.655211E-5 & 7.307688E-6 \\ -7.337804E-3 & -9.999732E-1 & -1.610292E-5 & -1.307002E-4 \\ -1.675308E-5 & -1.423457E-5 & 9.999108E-1 & -1.334777E-2 \\ 6.125366E-6 & -1.309519E-4 & 1.334783E-2 & 9.999109E-1 \end{bmatrix},$$

$$\tilde{A} = \begin{bmatrix} 1.000000 & -85.98243 & 20011.92 & -10194.38 \\ 4.733524 & 1.000000 & 19985.38 & 9807.223 \\ 0.000000 & 0.000000 & 2.000000 & -11.01783 \\ 0.000000 & 0.000000 & 39.48143 & 2.000000 \end{bmatrix}.$$

Test matrix II with parameter τ ($|\det(X) \ll \|X\|^2$)

$$A(\tau) = \begin{bmatrix} -3 & -87 & 3576\tau & 4888\tau \\ 5 & -3 & -88\tau & -1440\tau \\ 0 & 0 & 17 & -45 \\ 0 & 0 & 37 & 17 \end{bmatrix}$$

τ	Algorithm	e see (14)	E_A see (13)	E_P see (12)
1	0	3.32e-5	1.74e-7	4.42e-7
	1	**1.99e-3**	1.93e-6	6.39e-6
	2	1.52e-5	2.18e-7	2.10e-7
2^{-3}	0	1.37e-5	2.65e-7	4.94e-7
	1	**2.41e-4**	9.09e-7	1.85e-6
	2	1.50e-6	1.96e-7	5.12e-7
2^{-6}	0	7.54e-6	6.16e-7	5.69e-7
	1	7.94e-7	1.99e-7	2.60e-7
	2	1.68e-6	2.12e-7	7.49e-7

P and $\tilde{A} = P^T A P$ from algorithm 2 when $A = A(1)$.

$$X = \begin{bmatrix} 91.79044 & -124.2202 \\ -9.375406 & 19.85028 \end{bmatrix},$$

$$P = \begin{bmatrix} -9.907388E-1 & -1.318285E-1 & 3.216357E-2 & -4.818546E-3 \\ 1.356281E-1 & -9.640296E-1 & 2.282810E-1 & 1.181298E-2 \\ 4.287299E-3 & 1.861262E-1 & 8.120817E-1 & -5.530479E-1 \\ -4.807638E-3 & 1.364735E-1 & 5.360752E-1 & 8.330517E-1 \end{bmatrix},$$

$$\tilde{A} = \begin{bmatrix} 17.00000 & -1432.443 & -5568.100 & -2230.135 \\ 1.162349 & 17.00000 & 91.36795 & 824.0479 \\ 0.000000 & 0.000000 & -3.000000 & -236.8775 \\ 0.000000 & 0.000000 & 1.836391 & -3.000000 \end{bmatrix}.$$

Test matrix III with parameter τ (close eigenvalues)

$$A(\tau) = \begin{bmatrix} 7.001 & -87 & 39.4\tau & 22.2\tau \\ 5 & 7.001 & -12.2\tau & -36\tau \\ 0 & 0 & 7.01 & -11.7567 \\ 0 & 0 & 37 & 7.01 \end{bmatrix}.$$

τ	Algorithm	ε see (14)	E_A see (13)	E_P see (12)
1	0	5.85e-6	8.73e-7	8.34e-7
	1	3.50e-7	2.91e-7	2.67e-7
	2	4.69e-7	1.19e-7	2.48e-7
2^{-3}	0	6.12e-6	7.73e-7	1.30e-6
	1	6.12e-7	2.78e-7	2.36e-7
	2	6.18e-7	2.68e-7	8.03e-7
2^{-6}	0	4.61e-5	6.49e-7	7.89e-7
	1	3.20e-6	4.45e-7	4.05e-7
	2	2.83e-6	3.45e-7	5.01e-7

P and $\tilde{A} = P^T AP$ from algorithm 2 when $A = A(1)$.

$$X = \begin{bmatrix} 12581.73 & -2869.060 \\ 1218.421 & 1770.267 \end{bmatrix},$$

$$P = \begin{bmatrix} -1.000000E{+}0 & -1.293420E{-}4 & 6.833661E{-}5 & -4.892822E{-}5 \\ 1.293048E{-}4 & -9.999997E{-}1 & 1.150727E{-}4 & 5.055802E{-}4 \\ 6.830178E{-}5 & 1.152873E{-}4 & 1.000000E{+}0 & 4.074871E{-}4 \\ -4.902146E{-}5 & 5.055270E{-}4 & -4.076063E{-}4 & 9.999997E{-}1 \end{bmatrix},$$

$$\tilde{A} = \begin{bmatrix} 7.010000 & -87.01575 & -39.38432 & -22.17753 \\ 4.999070 & 7.010000 & 12.19859 & 36.00098 \\ 0.000000 & 0.000000 & 7.000999 & -11.75932 \\ 0.000000 & 0.000000 & 36.99190 & 7.000999 \end{bmatrix}$$

6. Conclusion

The test results in section 5 reveal that all three aglorithms are acceptable since Norm error measures E_A are tiny. Algorithm 1 and 2 have the advantage of keeping the real eigenvalues on the diagonals

at all times. The finer measure ϵ indicates that Algorithm 0 and Algorithm 1 in certain cases are inferior to Algorithm 2 but in other tests cases the roles of Algorithm 1 and 2 are reversed.

We find no reason to reject any of the methods and can give no perference.

7. References

[Ste,1976] G.W. Stewart, Algorithm 506 "HQR3 and EXCHNG: Fortran Subroutines for Calculating and Ordering the Eigenvalues of a Real Upper Hessenberg Matrix [F2]" , ACM TOMS Vol 2 No 3, September 1976 p275-280.

[Pa,1977] B.N.Parlett, "A program to swap diagonal blocks", UCB/ERL M77/66, November 1977.

[B&S,1972] Bartels, R.H., Stewart, G.W. Algorithm 432, Solution of the matrix equation AX+XB=C. Comm. ACM, vol.15, 820-826 (1972).

[Gant] Gantmacher, F.R., "Theory of Matrices, Vol I, (Chelsea, New York 1959).

[G,N,&vL,1979]
Golub, G.H, Nash, S.VanLoan, C.: A Hessenberg-Schur form for the problem AX+XB=C. IEEE Trans. Aut. Cont. AC-24, 909-913(1979)

[EIS,1976] Smith, B.T., et al; Matrix Eigensystem Routines - EISPACK Guide, Second edition. Leture Notes in Computer Science, N116 (Springer-Verlag, 1976).

Appendix A. Solving $A_1X - XA_2 = B$

When A_1 and A_2 are in standard form the inverse of the coefficient matrix can be expressed quite succinctly and safely. Let

$$A_i = \begin{bmatrix} \alpha_i & \beta_i \\ \gamma_i & \alpha_i \end{bmatrix}, \quad i=1,2, \quad \gamma_i \cdot \beta_i < 0.$$

Let $\delta = \alpha_1 - \alpha_2$, then the equations for solving X may be written as

$$(1) \quad \begin{bmatrix} C & \beta_1 I_2 \\ \gamma_1 I_2 & C \end{bmatrix} \cdot \mathbf{x} = \mathbf{b}; \quad \mathbf{x} = \begin{bmatrix} x11 \\ x12 \\ x21 \\ x22 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b11 \\ b12 \\ b21 \\ b22 \end{bmatrix}$$

where

$$(2) \qquad C = \begin{bmatrix} \delta & -\gamma_2 \\ -\beta_2 & \delta \end{bmatrix}, \quad C^2 = \begin{bmatrix} \delta^2 + \beta_2\gamma_2 & -2\delta\gamma_2 \\ -2\delta\beta_2 & \delta^2 + \beta_2\gamma_2 \end{bmatrix}.$$

Multiply (1) as indicated in order to make the coefficient matrix block diagonal,

$$(3) \qquad \begin{bmatrix} C^2 - \beta_1\gamma_1 & 0 \\ 0 & C^2 - \beta_1\gamma_1 \end{bmatrix} \cdot \mathbf{x} = \begin{bmatrix} C & -\beta_1 I_2 \\ -\gamma_1 I_2 & C \end{bmatrix} \cdot \mathbf{b}.$$

Now let

$$G = (C^2 - \beta_1\gamma_1)^{-1} = \begin{bmatrix} \tau & 2\delta\gamma_2 \\ 2\delta\beta_2 & \tau \end{bmatrix} / d$$

where

$$(4) \qquad \tau = \delta^2 + \beta_2\gamma_2 - \beta_1\gamma_1, \quad d = \tau^2 - (2\delta\beta_2)(2\delta\gamma_2) > 0,$$

and premultiply (3) by diag(G,G) to find

$$\mathbf{x} = \begin{bmatrix} G & 0 \\ 0 & G \end{bmatrix} \cdot \begin{bmatrix} C & -\beta_1 I_2 \\ -\gamma_1 I_2 & C \end{bmatrix} \cdot \mathbf{b}$$

$$(5) \qquad = \begin{bmatrix} G & 0 \\ 0 & G \end{bmatrix} \cdot \begin{bmatrix} \delta & -\gamma_2 & -\beta_1 & 0 \\ -\beta_2 & \delta & 0 & -\beta_1 \\ -\gamma_1 & 0 & \delta & -\gamma_2 \\ 0 & -\gamma_1 & -\beta_2 & \delta \end{bmatrix} \cdot \mathbf{b}\,,$$

$$= \frac{1}{d} \begin{bmatrix} \eta\delta & \psi\gamma_2 & -\tau\beta_1 & -2\delta\gamma_2\beta_1 \\ \psi\beta_2 & \eta\delta & -2\delta\beta_1\beta_2 & -\tau\beta_1 \\ -\tau\gamma_1 & -2\delta\gamma_1\gamma_2 & \eta\delta & \psi\gamma_2 \\ -2\delta\gamma_1\beta_2 & -\tau\gamma_1 & \psi\beta_2 & \eta\delta \end{bmatrix} \cdot \mathbf{b}\,,$$

where

$$\eta = \tau - 2\gamma_2\beta_2 = \delta^2 - (\beta_1\gamma_1 + \beta_2\gamma_2) > 0\,,$$
$$\psi = 2\delta^2 - \tau = \delta^2 + (\beta_1\gamma_1 - \beta_2\gamma_2)\,.$$

Inevitably (5) is Cramer's rule and $d = \det(A_1 \otimes I - I \otimes A_2)$ so that $d=0$ if and only $\alpha_1 = \alpha_2$, $\beta_1\gamma_1 = \beta_2\gamma_2$.

Remark. One step of iteration refinement may be needed if the structure matrix is ill-conditioned. We form the residual matrix $R=B-(A_1X-XA_2)$. If R is large relative to B, then using (5) again to solve for the correction matrix E_x from $A_1E_x-E_xA_2 = R$ and refine X by subtracting E_x from X.

Appendix B. Representing reflectors in form (6) of section 3

From (6) in section 3 we have

$$H \cdot \begin{bmatrix} -X \\ I_q \end{bmatrix} = \begin{bmatrix} C_2^T \\ 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} I_p & X \end{bmatrix} \cdot H^T = \begin{bmatrix} 0 & C_1 \end{bmatrix}.$$

Thus to represent the reflector(s) in 4.4 we need only to compute C_1 and C_2 using the formulae above. We skip the details of algebric manipulations and give the results below.

case 1×1: $C_1 = (sx)$, $C_2 = (-sx)$, where $sx = \text{sign}(x11)\cdot\sqrt{1+x11^2}$.

case 1×2: $C_1 = (sx)$, and

$$C_2 = \begin{bmatrix} -x11 & 1 \\ -x12 - \dfrac{1}{u3} & -\dfrac{x11}{u3} \end{bmatrix}, \quad \det(C_2) = sx,$$

where

$sx = -\text{sign}(x12)\cdot\sqrt{1+x11^2+x12^2}$,

if $-x12/sx \le 0.5$, then $u3 = x12+sx$; else $u3 = (1+x11^2)/(sx-x12)$.

case 2×1: $C_2 = (sx)$, and

$$C_1 = \begin{bmatrix} \dfrac{x21}{u1} & x11 - \dfrac{1}{u1} \\ 1 & x21 \end{bmatrix}, \quad \det(C_1) = sx,$$

where

$sx = \text{sign}(x11)\cdot\sqrt{1+x11^2+x21^2}$,

if $x11/sx \le 0.5$, then $u1 = sx - x11$; else $u1 = (1+x21^2)/(sx+x11)$.

case 2×2:

$$C_1 = \begin{bmatrix} x11 - \dfrac{sy - x22}{u1\cdot v2} & x12 - \dfrac{x21}{u1\cdot v2} \\ x21 - \dfrac{y3}{v2} & x22 - \dfrac{1}{v2} \end{bmatrix},$$

$$C_2 = \begin{bmatrix} -sx & 0 \\ y1 & -sy \end{bmatrix}, \quad \det(C_1) = \det(C_2) = sx\cdot sy,$$

where sx,u1,y1,y3,y2,sy,v2 are defined by

$$sx = \mathrm{sign}(x11)\cdot\sqrt{1+x11^2+x21^2}\ ,$$

If $x11/sx \le 0.5$, then $u1 = sx-x11$; else $u1=(1+x21^2)/(sx+x11)$,

$$y1 = -(x11\cdot x12+x21\cdot x22)/sx$$

$$y3 = (x12\cdot u1-x21\cdot x22)/(u1\cdot sx),$$

$$y2 = -x22-x21\cdot y3,$$

$$sy = -\mathrm{sign}(y2)\cdot\sqrt{1+y2^2+y3^2}\ ,$$

if $-y2/sy \le 0.5$, then $v2 = sy+y2$; else $v2 = (1+y3^2)/(sy-y2)$.

Appendix C. Listing of Fortran Subroutines

Subroutine SWAPB (Algorithm 1)
Subroutine SWAPB (Algorithm 2)
Subroutine HOUSE (used in Algorithm 2's SWAPB)
Subroutine EQUDI
Subroutine TXMXT

```
      SUBROUTINE SWAPB(T,P,N,J1,N1,N2,NT,NP)
      REAL T(NT,N),P(NP,N)
      INTEGER NP,NT,N,J1,N1,N2
C  GIVEN T IN SCHUR FORM SWAPB SWAPS ADJACENT DIAGONAL BLOCKS T1
C  AND T2 IN MATRIX T BEGINNING IN ROW J1 BY ORTHOGONAL SIMILARITY
C  TRANSFORMATIONS THAT PRESERVES THE SCHUR FORM OF T.  THE
C  DIMENSION OF BLOCK T1 IS N1 BY N1 AND T2 IS N2 BY N2. THE
C  PARAMETERS IN THE CALLING SEQUENCE ARE (STARRED PARAMETERS ARE
C  ALTERED BY THE SUBROUTINE)
C      *T      THE MATRIX WHOSE BLOCKS ARE BEING SWAPPED.
C      *P      THE ARRAY INTO WHICH THE TRANSFORMATIONS
C              ARE TO BE ACCUMULATED.
C      N       THE ORDER OF THE MATRIX T.
C      J1      THE POSITION OF THE BLOCKS.
C      N1      SIZE OF THE FIRST BLOCK.
C      N2      SIZE OF THE SECOND BLOCK.
C      NT      THE FIRST DIMENSION OF THE ARRAY T.
C      NP      THE FIRST DIMENSION OF THE ARRAY P.
C
C  METHOD:
C      ALGORITHM 1 OF "PROGRAMS TO SWAP DIAGONAL BLOCKS" WITH
C      SPECIAL FORMULA FOR THE DIAGONAL BLOCKS
C
C  SUBPROGRAMS:
C      TXMXT, EQUDI
C
C  INTERNAL VARIABLES:
C
      REAL D,A,S,Y,Z,U1,U2,U3,V1,V2,V3,Y1,Y2,Y3,Y4,W1,W2,W3,W4
      REAL T11,T22,T33
      REAL X(2,2),X11,X12,X21,X22
      REAL W(2,2),W11,W12,W21,W22
      REAL V(2,2),V11,V12,V21,V22
      REAL A1(2,2),A2(2,2)
      EQUIVALENCE (X(1,1),X11),(X(1,2),X12),(X(2,1),X21),(X(2,2),X22)
      EQUIVALENCE (W(1,1),W11),(W(1,2),W12),(W(2,1),W21),(W(2,2),W22)
      EQUIVALENCE (V(1,1),V11),(V(1,2),V12),(V(2,1),V21),(V(2,2),V22)
      INTEGER IZ,I,K,J1,J2,J3,J4
C
C SOLVE X FOR [I -X] [A1   0] [I  X] = [A1 T12]  BY CALLING TXMXT
C             [0  I] [0   A2] [0  I]   [0   A2]
C
      CALL TXMXT(T,N,J1,N1,N2,X,IZ,NT)
C
C IF IZ=0, A1 AND A2 ARE TOO CLOSE TO SWAP
C
      IF(IZ.EQ.0) GOTO 50
      K=N1+N1+N2-2
      J2 = J1+1
      J3 = J1+N1
      J4 = J3+1
      IF(N1.EQ.2) THEN
          A1(1,1)=T(J1,J1)
          A1(1,2)=T(J1,J2)
          A1(2,1)=T(J2,J1)
          A1(2,2)=T(J2,J2)
```

```
          ENDIF
          IF(N2.EQ.2) THEN
              A2(1,1)=T(J3,J3)
              A2(1,2)=T(J3,J4)
              A2(2,1)=T(J4,J3)
              A2(2,2)=T(J4,J4)
          ENDIF
          GOTO (10,20,30,40) , K
C
C N1=1, N2=1 : H = [  S   0 ] [ -X11   1 ], S = 1.0/SQRT(1+X11**2)
C                  [  0   S ] [ 1    X11 ]
C
 10       S=1.0/SQRT(1.0+X11*X11)
C
C PERFORM H*T*HT
C
          T11 = T(J1,J1)
          T22 = T(J2,J2)
          DO 12 I=J1,N
              W1 = T(J1,I)
              W2 = T(J2,I)
              Y1 = S*(-X11*W1 + W2)
              Y2 = S*( W1 + X11*W2)
              T(J1,I) = Y1
              T(J2,I) = Y2
 12       CONTINUE
          DO 14 I=1,J2
              W1 = T(I,J!)
              W2 = T(I,J2)
              Y1 = S*(-X11*W1 + W2)
              Y2 = S*(W1  + X11*W2)
              T(I,J1) = Y1
              T(I,J2) = Y2
 14       CONTINUE
C
C PERFORM P*HT
C
          DO 16 I=1,N
              W1 = P(I,J1)
              W2 = P(I,J2)
              Y1 = S*(-X11*W1 + W2)
              Y2 = S*( W1 + X11*W2)
              P(I,J1) = Y1
              P(I,J2) = Y2
 16       CONTINUE
C
C SWAP DIAGONAL ELEMENTS
C
          T(J2,J2)=T11
          T(J1,J1)=T22
          GOTO 50
C
C                      [ U1   0   0 ] [ -X11   1   0 ]
C N1=1, N2=2  :  H = [ U2  U3   0 ] [ -X12   0   1 ]
C                      [ 0    0  V1 ] [   1  X11 X12]
C
 20       S  = 1+X11*X11
```

```
          Y  = X12*X12
          V11= SQRT(S)
          V22= SQRT(1.0+Y/S)
          V21= 0
          V12= X11*(X12/V11)
          V1 = 1.0/(SQRT(S+Y))
          U1 = 1.0/V11
          U3 = 1.0/V22
          U2 = -(X11*X12*U3)/S
          T11 = T(J1,J1)
          J3  = J2+1
C
C PERFORM H*T*HT
C
          DO 22 I=J1,N
              W1 = T(J1,I)
              W2 = T(J2,I)
              W3 = T(J3,I)
              Y1 = -X11*W1 + W2
              Y2 = -X12*W1 + W3
              Y3 = W1 + X11*W2 + X12*W3
              T(J1,I) = U1*Y1
              T(J2,I) = U2*Y1+U3*Y2
              T(J3,I) = V1*Y3
22        CONTINUE
          DO 24 I=1,J3
              W1 = T(I,J1)
              W2 = T(I,J2)
              W3 = T(I,J3)
              Y1 = -X11*W1 + W2
              Y2 = -X12*W1 + W3
              Y3 = W1 + X11*W2 + X12*W3
              T(I,J1) = U1*Y1
              T(I,J2) = U2*Y1+U3*Y2
              T(I,J3) = V1*Y3
24        CONTINUE
C
C PERFORM P*HT
C
          DO 26 I=1,N
              W1 = P(I,J1)
              W2 = P(I,J2)
              W3 = P(I,J3)
              Y1 = -X11*W1 + W2
              Y2 = -X12*W1 + W3
              Y3 = W1 + X11*W2 + X12*W3
              P(I,J1) = U1*Y1
              P(I,J2) = U2*Y1+U3*Y2
              P(I,J3) = V1*Y3
26        CONTINUE
C
C T(J3,J3) = T11; CALL EQUDI WITH A2(2,2), V(2,2) TO GET A2'
C
          T(J3,J3) = T11
          CALL EQUDI(T,P,N,J1,A2,V,V11*V22,NT,NP)
          GOTO 50
C
```

```
C                            [ U1   0    0 ] [ -X11 -X21  1 ]
C N1=2, N2=1  :  H = [ 0    V1   0 ] [   1    0   X11]
C                            [ 0    V2  V3 ] [   0    1   X21]
C
30      S  = 1+X11*X11
        Y  = X21*X21
        W11= SQRT(S)
        W22= SQRT(1+Y/S)
        W12= 0
        W21= X11*(X21/W11)
        U1 = 1.0/(SQRT(S+Y))
        V1 = 1.0/W11
        V3 = 1.0/W22
        V2 = -(X11*X21*V3)/S
        W21= -W21
        D  = W11
        W11= W22
        W22= D
        T33= T(J3,J3)
C
C PERFORM H*T*HT
C
        DO 32 I=J1,N
            W1 = T(J1,I)
            W2 = T(J2,I)
            W3 = T(J3,I)
            Y1 = -X11*W1 - X21*W2 + W3
            Y2 = W1 + X11*W3
            Y3 = W2 + X21*W3
            T(J1,I) = U1*Y1
            T(J2,I) = V1*Y2
            T(J3,I) = V2*Y2+V3*Y3
32      CONTINUE
        DO 34 I=1,J3
            W1 = T(I,J1)
            W2 = T(I,J2)
            W3 = T(I,J3)
            Y1 = -X11*W1 - X21*W2 + W3
            Y2 = W1 + X11*W3
            Y3 = W2 + X21*W3
            T(I,J1) = U1*Y1
            T(I,J2) = V1*Y2
            T(I,J3) = V2*Y2+V3*Y3
34      CONTINUE
C
C PERFORM P*HT
C
        DO 36 I=1,N
            W1 = P(I,J1)
            W2 = P(I,J2)
            W3 = P(I,J3)
            Y1 = -X11*W1 - X21*W2 + W3
            Y2 = W1 + X11*W3
            Y3 = W2 + X21*W3
            P(I,J1) = U1*Y1
            P(I,J2) = U1*Y2
            P(I,J3) = V2*Y2+V3*Y3
```

```
36      CONTINUE
C
C T(J1,J1) = T33; CALL EQUDI WITH A1(2,2), W(2,2) TO GET A1'
C
        T(J1,J1) = T33
        CALL EQUDI(T,P,N,J2,A1,W,W11*W22,NT,NP)
        GOTO 50
C
C                    [ U1  0   0  0 ] [ -X11 -X21  1   0  ]
C N1=2, N2=2  : H =  [ U2 U3   0  0 ] [ -X12 -X22  0   1  ]
C                    [ 0   0  V1  0 ] [   1    0  X11 X12 ]
C                    [ 0   0  V2 V3 ] [   0    1  X21 X22 ]
C
40      CONTINUE
        D = X11*X22-X12*X21
        S = 1+X11*X11
        D = X22*X22+D*D
        Z = X12*X12
        R = X21*X21
        Y = S+Z
        W11 = SQRT(Y)
        W22 = SQRT(1.0+(D+R)/Y)
        W12 = 0.0
        W21 = (X11*X21+X12*X22)/W11
        Y = S+R
        V11 = SQRT(Y)
        V22 = SQRT(1.0+(D+Z)/Y)
        V21 = 0.0
        V12 = (X11*X12+X21*X22)/V11
        U1  = 1.0/V11
        U3  = 1.0/V22
        U2  = -V12/(V11*V22)
        V1  = 1.0/W11
        V3  = 1.0/W22
        V2  = -W21/(W11*W22)
        W21 = -W21
        Y   = W11
        W11 = W22
        W22 = Y
C
C PERFORM H*T*HT
C
        DO 42 I=J1,N
            W1 = T(J1,I)
            W2 = T(J2,I)
            W3 = T(J3,I)
            W4 = T(J4,I)
            Y1 = -X11*W1 - X21*W2 + W3
            Y2 = -X12*W1 - X22*W2 + W4
            Y3 =  W1 + X11*W3 + X12*W4
            Y4 =  W2 + X21*W3 + X22*W4
            T(J1,I) = U1*Y1
            T(J2,I) = U2*Y1 + U3*Y2
            T(J3,I) = V1*Y3
            T(J4,I) = V2*Y3 + V3*Y4
42      CONTINUE
        DO 44 I=1,J4
```

```
          W1 = T(I,J1)
          W2 = T(I,J2)
          W3 = T(I,J3)
          W4 = T(I,J4)
          Y1 = -X11*W1 - X21*W2 + W3
          Y2 = -X12*W1 - X22*W2 + W4
          Y3 =  W1 + X11*W3 + X12*W4
          Y4 =  W2 + X21*W3 + X22*W4
          T(I,J1) = U1*Y1
          T(I,J2) = U2*Y1 + U3*Y2
          T(I,J3) = V1*Y3
          T(I,J4) = V2*Y3 + V3*Y4
44      CONTINUE
C
C PERFORM P*HT
C
        DO 46 I=1,N
          W1 = P(I,J1)
          W2 = P(I,J2)
          W3 = P(I,J3)
          W4 = P(I,J4)
          Y1 = -X11*W1 - X21*W2 + W3
          Y2 = -X12*W1 - X22*W2 + W4
          Y3 =  W1 + X11*W3 + X12*W4
          Y4 =  W2 + X21*W3 + X22*W4
          P(I,J1) = U1*Y1
          P(I,J2) = U2*Y1 + U3*Y2
          P(I,J3) = V1*Y3
          P(I,J4) = V2*Y3 + V3*Y4
46      CONTINUE
C
C CALL EQUDI WITH A1, W TO GET A1', A2,V TO GET A2'
C
        CALL EQUDI(T,P,N,J1,A2,V,V11*V22,NT,NP)
        CALL EQUDI(T,P,N,J3,A1,W,W11*W22,NT,NP)
50      RETURN
        END
```

```
      SUBROUTINE SWAPB(T,P,N,J1,N1,N2,NT,NP)
      REAL T(NT,N),P(NP,N)
      INTEGER NP,NT,N,J1,N1,N2
C  GIVEN T IN SCHUR FORM SWAPB SWAPS ADJACENT DIAGONAL BLOCKS T1
C  AND T2 IN MATRIX T BEGINNING IN ROW J1 BY ORTHOGONAL SIMILARITY
C  TRANSFORMATIONS THAT PRESERVES THE SCHUR FORM OF T.  THE
C  DIMENSION OF BLOCK T1 IS N1 BY N1 AND T2 IS N2 BY N2.THE
C  PARAMETERS IN THE CALLING SEQUENCE ARE (STARRED PARAMETERS ARE
C  ALTERED BY THE SUBROUTINE)
C       *T      THE MATRIX WHOSE BLOCKS ARE BEING SWAPPED.
C       *P      THE ARRAY INTO WHICH THE TRANSFORMATIONS
C               ARE TO BE ACCUMULATED.
C       N       THE ORDER OF THE MATRIX T.
C       J1      THE POSITION OF THE BLOCKS.
C       N1      SIZE OF THE FIRST BLOCK.
C       N2      SIZE OF THE SECOND BLOCK.
C       NT      THE FIRST DIMENSION OF THE ARRAY T.
C       NP      THE FIRST DIMENSION OF THE ARRAY P.
C
C  METHOD:
C       ALGORITHM 2 OF "PROGRAMS TO SWAP DIAGONAL BLOCKS" WITH
C       SPECIAL FORMULA FOR THE DIAGONAL BLOCKS
C
C  SUBPROGRAMS:
C       TXMXT, EQUDI
C
C  INTERNAL VARIABLES:
C
      REAL X(2,2),U(4),UU(4),D,G,X11,X22,X12,X21,HALF,Y1,Y2,Y3
      REAL W(2,2),W11,W12,W21,W22
      REAL V(2,2),V11,V12,V21,V22,TEMP,T11,T22,T33
      REAL A1(2,2),A2(2,2)
      EQUIVALENCE (X(1,1),X11),(X(1,2),X12),(X(2,1),X21),(X(2,2),X22)
      EQUIVALENCE (W(1,1),W11),(W(1,2),W12),(W(2,1),W21),(W(2,2),W22)
      EQUIVALENCE (V(1,1),V11),(V(1,2),V12),(V(2,1),V21),(V(2,2),V22)
      INTEGER K
      HALF = 0.5
C
C SOLVE X FOR [I -X] [T1 T12] [I  X] = [T1  0 ]  BY CALLING TXMXT
C             [0  I] [0   T2] [0  I]   [0   T2]
C
      CALL TXMXT(T,N,J1,N1,N2,X,IZ,NT)
C
C IF IZ=0, A1 AND A2 ARE TOO CLOSE TO SWAP
C
      IF(IZ.EQ.0) GOTO 50
      K=N1+N1+N2-2
      J2 = J1+1
      J3 = J1+N1
      J4 = J3+1
      IF(N1.EQ.2) THEN
         A1(1,1)=T(J1,J1)
         A1(1,2)=T(J1,J2)
         A1(2,1)=T(J2,J1)
         A1(2,2)=T(J2,J2)
      ENDIF
```

```
      IF(N2.EQ.2) THEN
         A2(1,1)=T(J3,J3)
         A2(1,2)=T(J3,J4)
         A2(2,1)=T(J4,J3)
         A2(2,2)=T(J4,J4)
      ENDIF
      GOTO (10,20,30,40) , K
C
C 1,1 : H=I-UU*/D,  H [ X11 ] = [ S ], S=SIGN(X11)*SQRT(1+X11**2)
C                     [  1  ]   [ 0 ]
C
10    S= SIGN(SQRT(1.0+X11*X11),X11)
      T11  = T(J1,J1)
      T22  = T(J2,J2)
      U(1) = S - X11
      IF((X11/S).GT.HALF) U(1) = 1.0/(S+X11)
      U(2) = 1
      D    = U(1)*S
      CALL HOUSE(T,P,N,J1,U,2,D,NT,NP)
C
C SWAP DIAGONAL ELEMENTS
C
      T(J1,J1) = T22
      T(J2,J2) = T11
      GOTO 50
C
C 1,2 : [ 1 X11 X12 ] H = [ 0 0 S ], S=-SIGN(X12)*SQRT(1+X11**2+X12**2)
C
20    Y = 1.0+X11*X11
      S = SIGN(SQRT(Y+X12*X12),-X12)
      U(1) = 1
      U(2) = X11
      U(3) = X12 + S
      IF((-X12/S).GT.HALF) U(3) = Y/(S - X12)
      D = U(3)*S
      V11 = -X11
      V22 = -X11/U(3)
      V21 = 1.0
      V12 = -X12-1.0/U(3)
      T11 = T(J1,J1)
      CALL HOUSE(T,P,N,J1,U,3,D,NT,NP)
C
C T(J3,J3) = T11; CALL EQUDI WITH A2(2,2), V(2,2) TO GET A2'
C
      T(J3,J3) = T11
      CALL EQUDI(T,P,N,J1,A2,V,S,NT,NP)
      GOTO 50
C
C 2,1 : H [-X11] = [ S],        S = SIGN(X11)*SQRT(1+X11*X11+X21*X21)
C         [-X21] = [ 0]
C         [  1 ] = [ 0]
C
30    T33 = T(J3,J3)
      Y = 1.0+X21*X21
      S = SIGN(SQRT(Y+X11*X11),X11)
      U(1) = S - X11
      IF((X11/S).GT.HALF) U(1) = Y/(S+X11)
```

```
      U(2) = -X21
      U(3) = 1
      D    = U(1)*S
      W22  = X21/U(1)
      W11  = X21
      W12  = -X11+1.0/U(1)
      W21  = -1.0
      CALL HOUSE(T,P,N,J1,U,3,D,NT,NP)
C
C T(J1,J1) = T33; CALL EQUDI WITH A1(2,2), W(2,2) TO GET A1'
C
      T(J1,J1) = T33
      CALL EQUDI(T,P,N,J2,A1,W,S,NT,NP)
      GOTO 50
C
C 2,2 : H1 [-X11] = [ S],       S = SIGN(X11)*SQRT(1+X11*X11+X21*X21)
C          [-X21] = [ 0]
C          [  1 ] = [ 0]
C          [  0 ] = [ 0]
C
40    Y  = 1.0+X21*X21
      SX = SIGN(SQRT(Y+X11*X11),X11)
      U(1) = SX - X11
      IF((X11/SX).GT.HALF) U(1) = Y/(SX+X11)
      U(2) = -X21
      U(3) = 1
      D    = U(1)*SX
      CALL HOUSE(T,P,N,J1,U,3,D,NT,NP)
      TEMP = (T(J4,J1)*U(1)+T(J4,J2)*U(2)+T(J4,J3)*U(3))/D
      T(J4,J1) = T(J4,J1) - TEMP*U(1)
      T(J4,J2) = T(J4,J2) - TEMP*U(2)
      T(J4,J3) = T(J4,J3) - TEMP*U(3)
      Y1   = -(X11*X12+X21*X22)/SX
      Y3   = (X12*U(1)-X21*X22)/D
      Y2   = -X22-X21*Y3
C
C     H2 [Y1] = [Y1] , WERE Y = H1*[-X12], S = -SIGN(Y2)*SQRT(1+Y2**2+Y3**2)
C        [Y2] = [ S]               [-X22]
C        [Y3] = [ 0]               [  0 ]
C        [Y4] = [ 1]               [  1 ]
      Y   = 1.0+Y3*Y3
      SY  = SIGN(SQRT(Y2*Y2+Y),-Y2)
      UU(2) = SY+Y2
      IF (ABS(Y2/SY).GT.HALF) UU(2)=Y/(SY-Y2)
      UU(3) = Y3
      UU(4) = 1
      G = UU(2)*SY
      CALL HOUSE(T,P,N,J2,UU(2),3,G,NT,NP)
      TEMP = (UU(2)*T(J2,J1)+UU(3)*T(J3,J1)+UU(4)*T(J4,J1))/G
      T(J2,J1) = T(J2,J1) - TEMP*UU(2)
      T(J3,J1) = T(J3,J1) - TEMP*UU(3)
      T(J4,J1) = T(J4,J1) - TEMP*UU(4)
      W11 = X22 - 1.0/UU(2)
      W22 = X11 - (SY-X22)/(U(1)*UU(2))
      W12 = -X12+X21/(U(1)*UU(2))
      W21 = -X21+Y3/UU(2)
      V11 = -SX
```

```
      V22 = -SY
      V12 = Y1
      V21 = 0.0
      Y   = SX*SY
C
C CALL EQUDI WITH A1(2,2), W(2,2) TO GET A1', A2,V TO GET A2'
C
      CALL EQUDI(T,P,N,J1,A2,V,Y,NT,NP)
      CALL EQUDI(T,P,N,J3,A1,W,Y,NT,NP)
50    RETURN
      END

      SUBROUTINE HOUSE(T,P,N,J1,U,K,D,NT,NP)
      REAL T(NT,N),P(NP,N),U(K),D
      INTEGER K,J1,N,NT,NP
C
C THIS SUBROUTINE PERFORMS HOUSEHOLDER TRANSFORMATION ON T AND ACCUMULATE
C THE TRANSFORMATION IN P :
C       T  =   (I-UU*/D)T(I-UU*/D)
C       P  =            P(I-UU*/D),    WHERE U* STANDS FOR THE TRANSPOSE OF U.
C THE TRANSFORMATION BEGINS AT T(J1,J1). THE LENGTH OF U IS K.
C
C INTERNAL VARIABLES:
C
      REAL S,ZERO
      INTEGER I,J
      ZERO=0.0
C
C ROW MODIFICATION
C
      IF(D.EQ.ZERO) GOTO 100
      DO 30 J=J1,N
         S=0.0
         DO 10 I=1,K
10       S=S+U(I)*T(J1+I-1,J)
         S=S/D
         DO 20 I=J1,J1+K-1
20       T(I,J)=T(I,J)-S*U(I-J1+1)
30    CONTINUE
C
C COLUMN MODIFICATION
C
      DO 60 I=1,J1+K-1
         S=0.0
         DO 40 J=1,K
40       S=S+U(J)*T(I,J1+J-1)
         S=S/D
         DO 50 J=J1,J1+K-1
50       T(I,J)=T(I,J)-S*U(J-J1+1)
60    CONTINUE
C
C ACCUMULATION
C
      DO 90 I=1,N
         S=0.0
         DO 70 J=1,K
70       S=S+U(J)*P(I,J1+J-1)
```

```
      S=S/D
      DO 80 J=J1,J1+K-1
80    P(I,J)=P(I,J)-S*U(J-J1+1)
90    CONTINUE
100   RETURN
      END
```

```
      SUBROUTINE EQUDI(T,P,N,J1,A,W,DETW,NT,NP)
      REAL T(NT,N),P(NP,N),A(2,2),W(2,2),DETW
      INTEGER J1,N,NT,NP
C
C THIS SUBROUTINE PERFORMS A UNITARY TRANSFORMATION (A REFLECTION)
C TO MAKE THE DIAGONAL ELEMENTS IN W*A*W^-1 EQUAL AND PUT THE FINAL
C RESULT IN T.
C [-COS(Q)  SIN(Q) ] [ T11  T12 ] [-COS(Q)  SIN(Q) ]   [ T11' T12' ]
C [ SIN(Q)  COS(Q) ] [ T21  T22 ] [ SIN(Q)  COS(Q) ] = [ T21' T11' ]
C THE TRANSFORMATION IS ACCUMULATED IN (POSTMULTIPLED TO) P.  THE
C INPUT A MUST HAVE EQUAL DIAGONAL ELEMENT. HERE DETW IS THE
C DETERMINANT OF W.
C
C     F77 GENERIC FUNCTIONS:  ATAN, COS, SIN
C
C INTERNAL VARIABLES
C
      INTEGER I,J,J2
      REAL Q,U,S,C,Z,V1,V2,TEMP,ZERO,ONE,HALF
      REAL A11,A12,A21,A22,W11,W12,W21,W22
      A11 = A(1,1)
      A12 = A(1,2)
      A21 = A(2,1)
      A22 = A(2,2)
      W11 = W(1,1)
      W12 = W(1,2)
      W21 = W(2,1)
      W22 = W(2,2)
      ONE  = 1.0
      HALF = 0.5
      ZERO = 0.0
      J2=J1+1
C
C DETERMINE THE ANGLE Q
C
      Z = A21*W12*W22 - A12*W11*W21
      U = A12*(W11+W21)*(W11-W21) + A21*(W22-W12)*(W22+W12)
      IF(U+Z.EQ.Z) THEN
          Q = ATAN(1.0)
      ELSE
          Q = HALF*ATAN((Z+Z)/U)
      ENDIF
      S = SIN(Q)
      C = COS(Q)
C
C NEW DIAGONAL BLOCK
C
      Z = C/S
      V1 = ( A21*W22*(W22-W12*Z)-A12*W21*(W21-W11*Z))/DETW
      Z = S/C
      V2 = (-A21*W22*(W22+W12*Z)+A12*W21*(W21+W11*Z))/DETW
      Z = A12*A21
      IF(ABS(V1).GT.ABS(V2)) THEN
          V2 = Z/V1
      ELSE
          V1 = Z/V2
```

```
      ENDIF
C
C ROW MODIFICATION
C
      DO 10 J=J2+1,N
         TEMP    = -C*T(J1,J)+S*T(J2,J)
         T(J2,J)=  S*T(J1,J)+C*T(J2,J)
         T(J1,J)=  TEMP
10    CONTINUE
C
C COLUMN MODIFICATION
C
      DO 20 I=1,J1-1
         TEMP    = T(I,J1)*(-C)+T(I,J2)*S
         T(I,J2)= T(I,J1)*S+T(I,J2)*C
         T(I,J1)= TEMP
20    CONTINUE
      T(J1,J1) = A11
      T(J1,J2) = V1
      T(J2,J1) = V2
      T(J2,J2) = A11
C
C ACCUMULATION
C
      DO 30 I=1,N
         TEMP    = P(I,J1)*(-C)+P(I,J2)*S
         P(I,J2)= P(I,J1)*S+P(I,J2)*C
         P(I,J1)= TEMP
30    CONTINUE
40    RETURN
      END
```

```
      SUBROUTINE TXMXT(T,N,J1,N1,N2,X,IZ,NT)
      REAL T(NT,N),X(2,2)
      INTEGER N,NT,J1,N1,N2,IZ
C THIS SUBROUTINE SOLVES FOR N1 BY N2 MATRIX X IN T1*X - X*T2 = T12.
C T1 AND T2 BEGIN IN ROWS J1 AND J2 RESPECTIVELY, T12 IS THE UPPER
C TRIANGULAR PART BETWEEN T1 AND T2. THIS PROGRAM ASSUMES THE
C DIAGONALS OF T1 (T2) ARE EQUAL. THE PARAMETERS IN THE CALLING
C SEQUENCE ARE (STARRED PARAMETERS ARE ALTERED BY THE SUBROUTINE)
C     *T       INPUT MATRIX
C     N        THE ORDER OF THE MATRIX T
C     C        J1      THE POSITION OF THE BLOCKS.
C     N1       SIZE OF THE FIRST BLOCK.
C     N2       SIZE OF THE SECOND BLOCK.
C     *X       OUTPUT MATRIX
C     *IZ      OUTPUT INDICATOR: 1-X SOLVED SUCCESSFULLY,
C                                0-OVERFLOW MAY OCCUR.
C     NT       THE FIRST DIMENSION OF THE ARRAY T.
C
C  INTERNAL VARIABLES:
C
      REAL D,DEL,BET1,BET2,GAM1,GAM2,T1,T2,TAU,ETA,PHI,DSQ
      REAL P1,P2,P3,P4,P5,P6,P7,P8,P9
      INTEGER I,J,K
      ZERO=0.0
      IZ = 1
C
C  INITIALIZE
C
      DO 1 I=1,2
      DO 1 J=1,2
1     X(I,J)=ZERO
      J2 = J1+N1
      DEL=T(J1,J1)-T(J2,J2)
      B11=T(J1,J2)
      IF(N1.EQ.2) THEN
          J1P1 = J1+1
          BET1 = T(J1,J1P1)
          GAM1 = T(J1P1,J1)
          B21  = T(J1P1,J2)
      ENDIF
      IF(N2.EQ.2) THEN
          J2P1 = J2+1
          BET2 = T(J2,J2P1)
          GAM2 = T(J2P1,J2)
          B12  = T(J1,J2P1)
          IF(N1.EQ.2) B22=T(J1P1,J2P1)
      ENDIF
      K=N1+N1+N2-2
      GOTO (10,20,30,40) , K
C
C 1 BY 1:  APH1*X - X*APH2 = B11
C
10    IF(DEL.EQ.ZERO) THEN
          IZ = 0
      ELSE
          X(1,1) = B11/DEL
```

```
      ENDIF
      GOTO 50
C
C 1 BY 2:  APH1*[X11 X12] - [X11 X12]*[APH2 BET2]  = [B11 B12]
C                                      [GAM2 APH2]
C
20    D = DEL*DEL - BET2*GAM2
      IF(D.EQ.ZERO) THEN
            IZ = 0
      ELSE
            X(1,1) = (DEL *B11 + GAM2*B12)/D
            X(1,2) = (BET2*B11 + DEL  *B12)/D
      ENDIF
      GOTO 50
C
C 2 BY 1:  [APH1 BET1]*[X11] - [X11]*APH2  = [B11]
C          [GAM1 APH1] [X21]   [X21]         [B21]
C
30    D = DEL*DEL - BET1*GAM1
      IF(D.EQ.ZERO) THEN
            IZ = 0
      ELSE
            X(1,1) = ( DEL *B11 - BET1*B21)/D
            X(2,1) = (-GAM1*B11 + DEL *B21)/D
      ENDIF
      GOTO 50
C
C 2 BY 1:  [APH1 BET1]*[X11 X12] - [X11 X12]*[APH2 BET2] = [B11 B12]
C          [GAM1 APH1] [X21 X22]   [X21 X22] [GAM2 APH2]   [B21 B22]
C
40    DSQ = DEL*DEL
      T1  = BET1*GAM1
      T2  = BET2*GAM2
      TAU = DSQ + (T2 - T1)
      D   = TAU*TAU - 4*DSQ*T2
      IF(D.EQ.ZERO) THEN
         IZ = 0
      ELSE
         ETA = DSQ - (T1+T2)
         PHI = DSQ + (T1-T2)
         T2  = -(DEL+DEL)
         T1  = T2*BET1
         T2  = T2*GAM1
         P1  = ETA*DEL
         P2  = PHI*GAM2
         P3  = -TAU*BET1
         P4  = T1*GAM2
         P5  = PHI*BET2
         P6  = T1*BET2
         P7  = -TAU*GAM1
         P8  = T2*GAM2
         P9  = T2*BET2
         X(1,1) = (P1*B11+P2*B12+P3*B21+P4*B22)/D
         X(1,2) = (P5*B11+P1*B12+P6*B21+P3*B22)/D
         X(2,1) = (P7*B11+P8*B12+P1*B21+P2*B22)/D
         X(2,2) = (P9*B11+P7*B12+P5*B21+P1*B22)/D
CC
```

```
CC
CC COMPUTE RESIDUAL
CC
            R1 = B11 - (DEL*X(1,1)-GAM2*X(1,2)+BET1*X(2,1))
            R2 = B12 - (-BET2*X(1,1)+DEL*X(1,2)+BET1*X(2,2))
            R3 = B21 - (GAM1*X(1,1)+DEL*X(2,1)-GAM2*X(2,2))
            R4 = B22 - (GAM1*X(1,2)-BET2*X(2,1)+DEL*X(2,2))
CC
CC
CC PERFORM ONE ITERATION IF R* IS NOT SMALL COMPARED TO B*
CC
            T1 = ABS(B11)+ABS(B12)+ABS(B13)+ABS(B14)
            T2 = 0.0625*(ABS(R1)+ABS(R2)+ABS(R3)+ABS(R4))
            IF(T1+T2.NE.T1) THEN
                X(1,1) = X(1,1) + (P1*R1+P2*R2+P3*R3+P4*R4)/D
                X(1,2) = X(1,2) + (P5*R1+P1*R2+P6*R3+P3*R4)/D
                X(2,1) = X(2,1) + (P7*R1+P8*R2+P1*R3+P2*R4)/D
                X(2,2) = X(2,2) + (P9*R1+P7*R2+P5*R3+P1*R4)/D
            ENDIF
        ENDIF
50      RETURN
        END
```

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) University of California, Berkeley	2a. REPORT SECURITY CLASSIFICATION Unclassified
	2b. GROUP

3. REPORT TITLE

Programs to Swap Diagonal Blocks

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
CPAM report, June 1987

5. AUTHOR(S) (First name, middle initial, last name)

K. C. Ng and B. N. Parlett

6. REPORT DATE June 1987	7a. TOTAL NO. OF PAGES 38	7b. NO. OF REFS
8a. CONTRACT OR GRANT NO. N00014-85-K-0180	9a. ORIGINATOR'S REPORT NUMBER(S)	
b. PROJECT NO.		
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
d.		

10. DISTRIBUTION STATEMENT

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Mathematics Branch Office of Naval Research Washington, DC 20360

13. ABSTRACT

The real Schur form of a real square matrix is block upper triangular. We study techniques for performing orthogonal similarity transformations that preserve block triangular form but alter the order of the eigenvalues along the (block) diagonal.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6801

Security Classification

END
FILMED
5-89
DTIC